The Use of Emotional Intelligence in the Process of Teaching Software Engineering

Justo N. HIDALGO, Maria Eugenia DÍEZ, Pilar VÉLEZ
Department of Computer Science and Engineering
University Antonio de Nebrija
c/Pirineos, 55 MADRID
jhidalgo@nebrija.es, mdiez@nebrija.es, pvelez@nebrija.es

KEYWORDS: *emotional intelligence, book of knowledge, software engineering, engineering-based sociology*

ABSTRACT: *Integration between the university environment and industrial world requires to the students a mastering of abilities which are beyond those purely technical. Qualities like leadership, stress control, ethics or communication skills are now essential for a professional to succeed in multidisciplinar teams and mission-critical projects. Engineering departments are now realizing of this necessity, and are creating new subjects related to psychology, sociology and enterprise economics, which are becoming of great importance in the program's curriculum. This paper proposes the use of emotional intelligence techniques not only as a final aim, but also as a tool for the teacher to use; the way to achieve this goal is by applying emotional intelligence skills when explaining the technical subjects. This way, the engineering student can realize of the usefulness of these techniques, since the teacher is consciously applying them. To defend this thesis, the main characteristics for an adequate teaching and learning of technical subjects are shown, how traditional establishments do not allow for a complete understanding of the set of themes, and how emotional intelligence, obviously starting from a complete knowledge of the subject to be treated, allows the teacher to deepen into the student's capabilities, motivating and leading them to a better profit of the class. This paper is divided in three main parts. The first one will study the intrinsic characteristics of an engineering titulation, and which are the main goals that a student must obtain from the subjects. The second chapter sustains a theoretical thesis for general application of emotional intelligence into the teaching of engineering courses. At last, the third part shows how an application of these ideas was built when teaching "Software Engineering", a course from the titulation of Computer Science in a spanish university.*

1 INTRODUCTION

Engineering work is always known to have many characteristics and requirements beyond those technical usually learnt at unversity. But it has not been until a few years ago when engineering departments have started to consider those needs as proper knowledge to be taught as subjects. Nowadays, courses such as Economics for Engineers, Control Production, Social Engineering or Emotional Intelligence skills are vital for students to successfully behave in a corporate environment, or understand how to understand what a client exactly wants in the requirements phase.

These courses help students to comprise the theoretical basics of emotional intelligence and economics. However, these subjects are usually taught by experts on the fields, such as economists, psychologists, sociologists, and so on. This is needed, for they are the ones who can better explain each of the concepts. Nevertheless, they are not engineers, and are not usually able to map the theoretical ideas behind each course to applications of real-world projects or engineering matters. Of course, one could always find an expert able to achieve this challenge, but what this article upholds is the generality of it. Engineers should be taught peripheral knowledge –such as emotional intelligence, economics, or managing abilities- in relation to the type of work they will have to successfully apply in the future.

One option would be that engineers teach those courses; that way, examples, activities and assignments will be oriented to what students will need in the future. However, they would not in general be able to adequately explain the theoretical basis and the underlying concepts of each subject. Engineers should focus on teaching technical or managerial areas; what this article proposes is that these teachers must also apply emotional intelligence techniques in most technical subjects –at least, the ones taught in third, fourth or fifth year-. Students will learn the theoretical basis of these conceptos in the specific subjects, but will learn to apply them to their daily professional life in the rest of subjects they have.

2 SOFTWARE ENGINEERING’S EDUCATIONAL NEEDS

Software Engineering is one of the youngest engineering titles in the world. Initially defined in 1968 [1], it has become a very important field since software has become an almost ubiquitous resource for every human activity.

However, software engineering is also one of the most difficult areas to work with, since it is not based on physical laws, but on mathematical, logic and even human-invented laws. This fact makes its mastering a very challenging task. Phases such as Requirements or Aceptance tests are proven to be very difficult, and many years of research have been spent trying to automatize them. The result is clear: these are human activities, and practicioner’s capabilities such as empathy, stress control, or assertivity are keypoints to master in order to succeed.

Current software engineering world is a hectic place where people are still the most important value. The problem is that uncertainty and stress are increasing factors [2][3]. Besides, professionals are asked to be skillful at teamwork, to be enthusiastic 24 hours a day, and have enough self-control to handle the multiple situations they must handle because of the multidisciplinary and heterogeneous environments they work at. Being able to master these abilities will lead to an improved intercommunication, an, above all, an increasing productivity.

Engineering schools must teach their students how to respond to these new challenges. As [Anita Harris] puts it, if an engineer is always asked to “calm down” by other colleagues, is never listened, or never chosen to be in a team, it does not matter how technically able they can be.

In [4], a study realized by the Business/Higher Education Round Table, Australia, in 1992, is shown in which both universities and industries priorize what they want from students and professionals. While universities give more importance to theoretical and general knowledge, more relevance is given by industry to interpersonal skills and specialized knowledge. Even though universities should keep focusing primarily on teaching students the basics of what they will have to apply in the future, they have also some responsibility on educating them in a holistic way [5] [6].

3 EMOTIONAL INTELLIGENCE IN SOFTWARE ENGINEERING

As it has been discussed previously, software engineering practicioners face many different challenges in their professional daily life that are not solved just by applying technical knowledge.

A software engineering project should always be led by a specific software development process, which assures enough quality and ordering of activities. Processes such as Unified Process, Agile Programming methodologies, ISO/IEC 12207 or Metrica v3 all look for sequencing a series of activities so that, in every moment, project managers, technical managers and developers might know what they are doing, how they should be doing it, and how long it must take them. All of them are evolutions of the waterfall lifecycle model, which had six basic phases: requirements, analysis, design, implementation, and testing. Table 1 shows what emotional competencies –based on [7], [8] and [9]- are required by each phase.

Table 1: Mapping between SW Engineering phases and Emotional Intelligence Components

Requirements	○ Personal • Emotional awareness • Self-control • Trustworthiness • Adaptability • Innovation • Achievement drive • Commitment • Lateral thinking ○ Social • Understanding others • Service orientation • Political awareness • Influence • Communication • Conflict management • Collaboration and cooperation
Analysis	○ Personal • Accurate self-assessment • Self-confidence • Conscientiousness • Adaptability • Innovation • Achievement drive ○ Social • Service orientation • Influence • Communication • Conflict management • Leadership • Team capabilities
Design	○ Personal • Accurate self-assessment • Self-confidence • Trustworthiness • Conscientiousness • Adaptability • Innovation • Achievement drive • Initiative • Lateral thinking ○ Social • Service orientation • Influence • Communication • Conflict management • Leadership • Change catalyst • Team capabilities
Implementation	○ Personal • Accurate self-assessment • Self-confidence • Trustworthiness • Conscientiousness • Adaptability

	• Innovation • Achievement drive • Developing others • Service orientation • Lateral thinking o Social • Communication • Conflict management • Change catalyst • Collaboration and cooperation • Team capabilities
Testing	o Personal • Accurate self-assessment • Self-control • Trustworthiness • Conscientiousness • Achievement drive • Optimism o Social • Service orientation • Communication • Collaboration and cooperation • Team capabilities

One by one, each of the abilities will be revised, focusing on how each one is required for each software product development phase.

3.1 Emotional awareness

The requirements' phase demands for the software engineer to communicate with the customer so that the system to be built is adequately understood by both parts, so that all functional and non-functional system requirements are defined. This phase is also used to identify the users which will operate the software once it is constructed. Finally, risk management is started in this phase, so to minimize the possible problems the project can have from the beginning.

This is the most emotionally-related engineering phase, for the keypoint here is the personal intercommunications skills. Recognizing one's emotions and their effects is vital when talking to a client. An engineer should think twice before answering a customer's enquiry at requirement's phase, because they are both signing an implicit contract.

3.2 Accurate self-assessment and self-confidence

Even though this ability is important in every single phase of a software project, knowing and accepting their own limits is essential for engineers when having to apply their knowledge in more lonely phases, such as it usually happens when analyzing, designing or developing software. Deciding whether one can handle a specific situation must be realized as soon as possible in a project; letting time goes by might in some cases help solve that specific problem, but most of the times it will only mean wasting more resources and time.

3.3 Self-control

Keeping one's emotions in check is also part of the business when eliciting requirements. In many situations, clients push the software providers in orden to obtain more benefits at the same price. Being able to control their own feelings, and applying other emotional abilities such as conflict management and influence, will be the best way to succeed.

This ability is also very important when an engineer has to test other professional's software. Patience is important to understand what others have designed and developed, but also, the fact that testing is the last phase before the end of the project –or the end of a particular iteration of the whole project- means more pressure for practicioners.

3.4 Trustworthiness and conscientiousness

Every software engineer must have a code of ethics, such as IEEE/ACM's one [10], and must accomplish it even if it means losing some job. At requirement's phase, promising things to the customer seems an easy way for the provider to be assigned the project, but even "little lies" such as offering functionalities that are not yet built can affect the overall quality of the system. At analysis and design phase, time pressure can force engineers to make some decisions which affect the overall quality of the system; while some of them only affect to characteristics such as escalability or reuse, professionals must take special care not to decrease the reliability of the product they are creating.

3.5 Adaptability

Adaptability is defined in [7] as "flexibility in handling change". Current software processes such as Unified Process or eXtreme Programming are naturally based in that concept, using iterative and incremental models –initially defined by the spyral lifecycle model from Boehm-. Clients do not have to know exactly what they want at the beginning of the project. The idea is to keep on meeting with them iteratively so that the system grows up systematically. To achieve that, software engineers must organize their work so that adding or modifying functionality does not affect radically the project plan.
Also, engineers in charge of the analysis, design and implementation phases must be ready for change. New, changing or canceled requirements will affect the utopic development of the project as it was initially planned.

3.6 Innovation, initiative and optimism

These characteristics are vital, and should be requirements for first-year students of engineering. As it has been said previously, this is a very young area and changes, technologic advances, ... , are unavoidable. Understanding these components, accepting them from the beginning of every project and taking profit from them, will improve the chances that it will finish successfully. Besides, even though innovation might increase the number of risks a project might suffer, it also gives more chances for professionals to learn and experience.

3.7 Achievement drive

Every project must be completed accomplishing a set of requirements, in a given time and with a limited amount of resources. A good engineer must obtain the best results taking into account these constraints, with the highest quality standards possible. If time or resources do not allow for perfect completion, customer and provider should sit down again and discuss which functionalities are more prioritary, but quality should not be put on jeopardy. This ability must be also accomplished in the rest of the phases. Every engineer should strive for excellence every single minute.

3.8 Commitment

Requirements phase requires engineers with a high degree of commitment towards the provider's goals, and above all, oneselves. Clients might have different needs, some of which might not be compatible with the ones from the provider. Finding adequate commitments, always based on the previously described engineering ethics, is another requirement for a software professional.

3.9 Service orientation

This is the summary of what a software project must be. Understanding, estimating, and, above all, being able to meet what a customer needs. An engineer must always follow this path, for it does not matter how well designed, developed or deployed a product has been realized, if it does not satisfy the customer, everything is worthless.

3.10 Political awareness
Understanding who the engineer is talking to, what each specific customer can decide or not, how important the project is inside the customer's organization, ... These are important facts that a provider must try to know in order to better serve the customer's and their own benefits. Political issues should not affect directly the quality of the project, but they are very important at the requirements' phase, so that the engineer in charge can understand the structure of the organization, and select the ones who can better answer each of the questions or necessities.

3.11 Influence
As it has been discussed before, an engineer must not always agree with what the customer wants. Because of ethical, functional or timely issues, the provider must sometimes persuade the client, and different techniques can be used to, at least, be able to discuss that particular item with them.

3.12 Communication
Communication skills when meeting with a client to decide which functionalities must a new and, usually, innovative system, is required. Being able to create effective channels of communication among different parties involved in the project implies more data and information, and from better sources, which therefore will be transformed in better requirements.

3.13 Conflict management
Customers pay to get what they want, but sometimes they want too much. On the other side, providers are paid to create a useful software system, but sometimes they do not care that much of what the customer really wants. In the middle of these two radical points, there are many situations in which a conflict may appear. Engineers must cope with these confrontations, and handle them while keeping in mind the common goal.

3.14 Leadership and change catalyst
Software engineering projects are realized by multidisciplinary teams composed by heterogeneous professionals, with different specializations and goals. That heterogeneity does not just happen because a software project mixes together customers and providers, or technicians with graphic designers, but also among engineers, some of whom are very specialized, such as system administrators, database experts, and so on, but others are generalists, such as project managers. Leaders of the project must be able to understand all different motivations and get everybody to understand and follow a single path with enthusiasm. Leaders do not need to be appointed in a hierarchical way, but it happens naturally. An ideal project is the one in which project managers are also project leaders. If not, after a period of time, real leaders will appear, but in medium-to-large-size projects, it would probably be too late.

3.15 Collaboration, cooperation and team capabilities
Very related to the previous ability, software projects' degree of success is proportional to the ability of the team to work together. This means that every engineer should have communication skills, teams should promote cooperation among all the members, but also to organize information and resources so that every single need can be easily searched and found.

3.16 Lateral Thinking

Due to the layer-based architecture that most software applications are designed, engineers face many challenges when finding that a logical solution can not be properly implemented in a particular system or framework.

Furthermore, treating with customers and coming to a conclusion which benefits both parts is not an easy task, and sometimes logical steps are not the best way to achieve the best result. Lateral thinking [7] is the way to find new steps which might not have been found in a logical flow tree.

4 TEACHING SOFTWARE ENGINEERING USING EMOTIONAL INTELLIGENCE

The department of Computer Science and Engineering of University Antonio de Nebrija has been working these last two academic courses in order to teach and apply emotional intelligence techniques. Starting with a yearly cycle of seminars on Emotional Intelligence for Software Engineers [11], the following step has been to apply Emotional Intelligence and Parallel Thinking in the process of teaching software engineering.

Many good teachers are already applying these techniques in different engineering courses, many of them without even knowing it; the main reason is that these teachers own different emotional intelligence abilities, so that their use is implicit. What this article proposes is a series of advices which can be directly used when teaching. Even though it might be very difficult to develop all of the following ideas in a single semester, instructors should find the ones which they find more important for their particular groups of students.

Table 2 is the keypoint of this chapter. The first column describes a set of actions which can be applied to a software engineering course. The second column lists each of the emotional intelligence abilities to be used by students, either explicitly or implictly. Third column, at last, shows the characteristics that the instructor must master in order to successfully develop that particular action.

The course is composed by the following parts: (1) several theoretical sessions in which software engineering concepts are introduced and explained; (2) a team-based case in which students must understand, discuss, and design an industry-size project; and (3) another project which students must analyze, design and implement, with a final demo in front of a customer.

4.1 Theoretical Sessions

Theoretical sessions should focus on the following actions:

1. Teaching basic and advanced theoretical concepts: this is the classic ideal of a session, but with must be always performed with students' participation.
2. Helping students to make their own decisions, and maintain them throughout the process: starting with little cases used in different topics throughout the course should help students to analyze how decisions taken at the beginning of a project affect dramatically the rest of the development.
3. Making students start to apply critical appraisal on their own and others' productions.
4. Improving students' self-learning. All information is stored in a web page, with labs, assignments, links to advanced resources, and so on. Students attendance is not imposed.

4.2 Case Study

The case study is explained and developed during the first third of the course. Students are forced to understand concepts which have not been explained before –p.e. during couse 2003/04 the case study was based on a bank-account aggregation tool implemented for a multinational bank-, just as it happens when a provider is introduced to a new customer. Also, students must perform different roles during the case study, starting as part of the bank's organization, but then acting as one provider which desires to obtain the project. Besides, students must produce different presentations and technical documents.

4.3 Team Project

During the rest of the course, students must fully analyze, design, develop, test and show a small project in which the importance resides on the use of a software process, such as Unified Process, ISO/IEC 12207 or eXtreme Programming. Students must work in teams in order to successfully achieve and supply all of the requirements initially discussed with the client – performed by the teacher or by some other professional-, produce the different artifacts or documents as required by the selected process, and, finally, creating a demo presentation in front of the same client. Students are responsible for creating the project plan, and deciding what milestones there will be.

Table 2: Course actions and their related emotional intelligence abilities

Action	Subaction	E.I. skills for students	E.I. skills for teachers
Theoretical sessions	Use of little case studies for each topic	- Conscientiousness - Adaptability - Commitment - Understanding others - Service orientation - Political awareness - Communication - Conflict management - Lateral thinking	- Self-confidence - Self-control - Developing others - Conflict management - Leadership - Change catalyst
Case study	Case study based on real project, meeting different actors.	- Emotional awareness - Innovation, initiative and optimism - Commitment - Service orientation - Political awareness - Conflict management	- Self-control - Understanding others - Communication - Accurate self-assessment - Self-confidence - Leadership - Conflict management
	Students must produce a project plan, acting as external providers	- Accurate self-assessment - Self-confidence - Trustworthiness - Innovation - Commitment - Understanding others - Service orientation - Influence - Team capabilities	- Emotional awareness - Achievement drive - Understanding others - Developing others - Political awareness
	Students must prepare a presentation of the solution to the CEO	- Accurate self-assessment - Self-confidence - Self-control - Trustworthiness - Conscientiousness - Optimism - Service orientation - Political awareness - Influence - Communication	- Accurate self-assessment - Self-control - Understanding others - Developing others - Leveraging diversity - Political awareness

Team Project	Project plan + meetings with customer	- Accurate self-assessment - Self-confidence - Self-control - Conscientiousness - Adaptability - Innovation - Achievement drive - Initiative - Understanding others - Developing others - Influence - Communication - Conflict management	- Self-confidence - Innovation - Initiative - Developing others - Leveraging diversity - Communication
	Iterative and incremental focus	- Accurate self-assessment - Adaptability - Achievement drive - Understanding others - Developing others - Service orientation - Communication - Conflict management - Leadership - Change catalyst - Collaboration and cooperation - Team capabilities - Lateral thinking	- Understanding others - Developing others - Leveraging diversity - Communication - Conflict management - Collaboration and cooperation
	Pre-arranged delivery date (goal-based evaluation)	- Accurate self-assessment - Self-confidence - Trustworthiness - Achievement drive - Initiative - Service orientation - Conflict management - Leadership - Team capabilities	- Self-control - Developing others - Influence - Conflict management - Collaboration and cooperation
	Demo presentation	- Emotional awareness - Accurate self-assessment - Self-confidence - Self-control - Conscientiousness - Adaptability - Optimism - Political awareness - Influence - Communication	- Self-control - Understanding others

5 CONCLUSIONS AND FUTURE WORK

This is a long-term project which tries to create a Book of Knowledge of Emotional Intelligence in Software Engineering. During these last two academic courses, the authors have been applying emotional intelligence and lateral thinking skills both explictly and implictly in software engineering courses, so that they can also be evaluated to students and instructors. It is important to notice the difference between applying these characteristics unknowingly and explictly. Some teachers might be emotionally able, and apply some, or all of these characteristics without thinking about it; nevertheless, teachers and instructors do not usually care about these elements, and believe that students will learn about them whenever they start to work in industry. This article defends their use from the beginning of, at least, third year of engineering.

The use of these techniques has improved the degree of success of engineering students, and their involvement in industry.

However, there are some drawbacks that are not yet solved: first of all, emotional intelligence skills should only be used explictly by teachers when they master the subject. Novice teachers should better get more experience before applying these techniques. Secondly, all techniques are not suited for every single student or group, and special care should be taken in order to avoid an extreme homogeneization of how students must behave. Actually, the ability of "understanding others" is vital for a teacher to master.

As future work, new projects and case studies must be produced. In order to improve them, participation of industry professionals and companies should be welcome. Their experience on real projects, tempered by instructors' experience would definitely launch the project. Besides, new applications of each emotional intelligence ability are to be found.

ACKNOWLEDGEMENTS

The authors would like to thank Colin Grindley for his support throughout this project, and his ideas about lateral thinking.

REFERENCES

[1] PETERS, L. 2003. *Educating Software Engineering Managers.* Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), pp. 78-85. 1093-0175/03

[2] AHTEE, T. 2003. Inspections and Historical Data in Teaching Software Engineering Project Course. Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), pp. 288-294. 1093-0175/03

[3] AL-KHATIB, W. G., BUKHRES, O. DOUGLAS, P. 1995. *An empirical study of skills assessment for software practioners*. Information-Sciences Applications, 4(2):83–118, 1995.

[4] TELES, V.M., DE OLIVEIRA, C. E. T. 2003. *Reviewing the Curriculum of Software Engineering Undergraduate Courses to Incorporate Communication and Interpersonal Skills Teaching.* Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), pp. 158-165. 1093-0175/03

[5] ELLIS, H. J. C., MEAD, N. R., MORENO, A. M., SEIDMAN, S. B., 2003. *Industry/University Software Engineering Collaborations for the Successful Reeducation of Non-Software Professionals.* Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03), pp. 44-51. 1093-0175/03

[6] LAMP, J., KEEN, C., URQUHART, C., 1996. *Integrating Professional Skills into the Curriculum*. Proceedings of the First Australasian Conference on Computer Science Education, Sydney, Australia, pp. 309-316, July 1996.

[7] GOLEMAN, D. 1997. *Emotional Intelligence. Why it can matter more than IQ*. Bantam. 0553375067.

[8] GOLEMAN, D. 2000. *Working with Emotional Intelligence.* Bantam. 0553378589

[9] SEGAL, J. S. 1997. *Raising your Emotional Intelligence: A Practical Guide*. Henry Holt & Company. 0805051511.

[10] *IEEE Code of Ethics*. Available from web: <URL: http://www.ieee.org/portal/index.jsp?pageID=corp_level1&path=about/whatis&file=code.xml&xsl=generic.xsl>

[11] *I Cycle on Emotional Intelligence for Software Engineers*. Poster available from web: <URL: http://www.nebrija.es/~jhidalgo/events/IE/cartel.pdf>